04 | The Google File System （二）： 如何应对网络瓶颈？

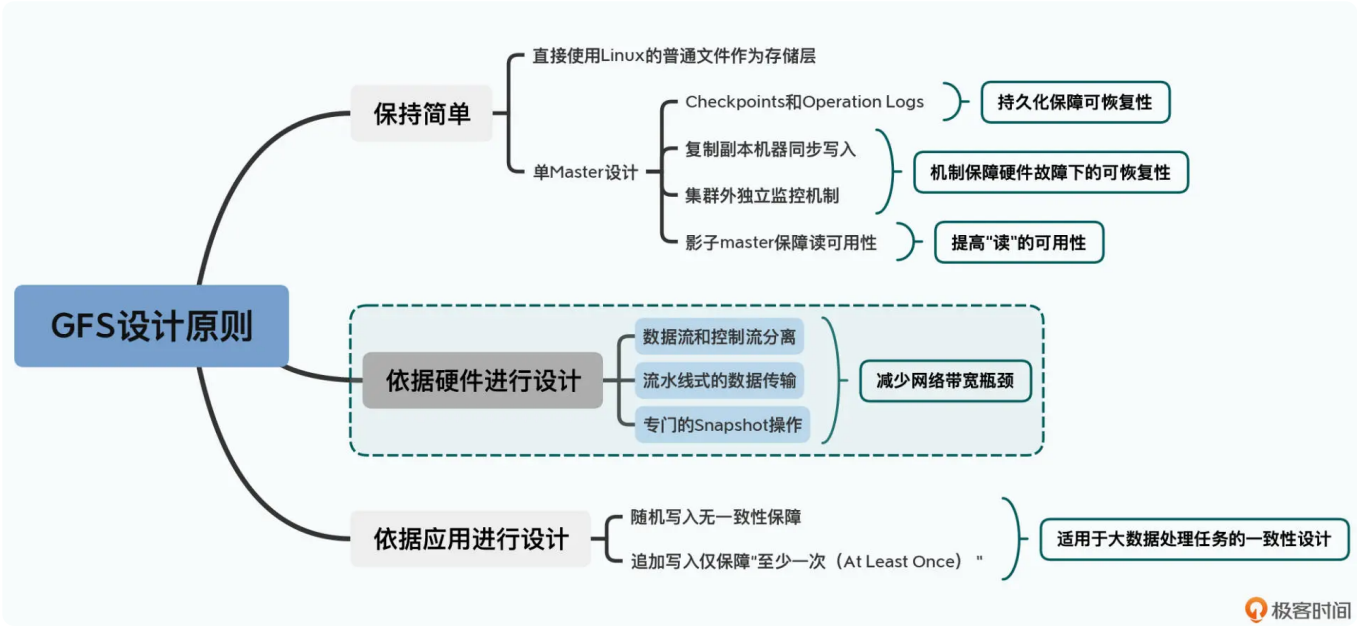
2021-09-27 徐文浩

《大数据经典论文解读》 课程介绍 >

讲述：徐文浩

时长 15:31 大小 14.22M

你好，我是徐文浩。今天这一讲，我们接着来学习 GFS 论文中第二个重要的设计决策，也就是根据实际的硬件情况来进行系统设计。



大数据系统本就是为“性能”而生的，因为单台服务器已经满足不了我们的性能需要。所以我们需要通过搭建成百上千台服务器，组成一个大数据集群。然而，上千台服务器的集群一样有来自各种硬件性能的限制。

在单台服务器下，我们的硬件瓶颈常常是硬盘。而到了一个分布式集群里，我们又有了一个新的瓶颈，那就是**网络**。

那么在这一讲里，我们就来看看网络层面的硬件瓶颈，是如何影响了 GFS 的设计的。在学完这一讲之后，希望你能够理解，**任何一个系统设计，都需要考虑硬件性能**。并且学会在对自己的设计进行评估的时候，能够寻找到系统的硬件瓶颈在哪里。

GFS 的硬件配置

不知道你有没有想过，2003 年的 GFS 是跑在什么样的硬件服务器上的呢？论文的第 6 部分还真的透露了一些信息给我们。Google 拿来做微基准测试（Micro-Benchmark）的服务器集群的配置是这样的：

> 19 台服务器、1 台 master、2 台 master 的只读副本、16 台 chunkserver，以及另外 16 台 GFS 的客户端；
>
> 所有服务器的硬件配置完全相同，都是双核 1.45 GHz 的奔腾 3 处理器 + 2GB 内存 + 两块 80GB 的 5400rpm 的机械硬盘 + 100 Mbps 的全双工网卡。
>
> 然后把所有的 19 台 GFS 集群的机器放在一台交换机上，所有的 16 台 GFS 的客户端放在另外一台交换机上，两台交换机之间通过带宽是 1Gbps 的网线连接起来。

而 Google 跑在内部实际使用的真实集群，虽然论文里没有给出具体的硬件配置，但我们也可以反向推算一下。论文第 6 部分有一个 Table 2，里面有 A 和 B 两个集群。根据表格里面的数据可以计算得出，里面的 A 集群平均每台 chunkserver 大约有 200GB 的硬盘，每台 chunkserver 需要的 Metadata（元数据）大约是 38MB。而里面的 B 集群则是 800GB 的硬盘，以及 93MB 的 Metadata。这样看起来，除了可以多插几块硬盘增加一些存储空间之外，前面测试集群的硬件配置完全够用了。

集群	A	B
chunkserver数量	342	227
可用硬盘空间	72TB	180TB
平均每台chunkserver的硬盘空间	~200GB	~800GB
chunkserver上的Metadata	13GB	21GB
平均每台chunkserver的Metadata	38MB	93MB

极客时间

以上数据可以从论文6.2部分的Table 2中计算得出

在这个硬件配置中，5400 转（rpm）的硬盘，读写数据的吞吐量通常是在 60MB/s~90MB/s 左右。而且我们通常会插入多块硬盘，比如集群 B，就需要 10 块 80GB 的硬盘，这样就意味着整体硬盘的吞吐量可以达到 500MB/s 以上。但是，100Mbps 的网卡的极限吞吐率只有 12.5MB/s，这个也就意味着，当我们从 GFS 读写数据的时候，**瓶颈就在网络上**。

那么下面，我们就来看一看针对这样的硬件瓶颈，GFS 都做了哪些针对性的设计。

GFS 的数据写入

我们先来看看一个客户端是怎么向 GFS 集群里面写数据的。在上一讲里，我带你了解了一个 GFS 客户端怎么从集群里读取数据。相信你学完之后会觉得特别简单，感觉也就是个几千行代码的事儿。不过，写文件可就没有那么简单了。

实际上，读数据简单，是因为不管我们有多少个客户端并发去读一个文件，读到的内容都不会有区别，即使它们读同一个 chunk 是分布在不同 chunkserver。我们不是靠在读取中做什么特殊的动作，来保障客户端读到的数据都一样。**“保障读到的数据一样”这件事情，其实是在数据写入的过程中来保障的。**

写入和读取不同的是，读取只需要读一个 chunkserver，最坏的情况无非是读不到重试。而写入，则是同时要写三份副本，如果一个写失败，两个写成功了，数据就已经不一致了。

那么，GFS 是怎么解决这样的问题的呢？下面我就带你来看一下，GFS 写入数据的具体步骤。

第一步，客户端会去问 master 要写入的数据，应该在哪些 chunkserver 上。

第二步，和读数据一样，master 会告诉客户端所有的次副本（secondary replica）所在的 chunkserver。这还不够，master 还会告诉客户端哪个 replica 是“老大”，也就是主副本（primary replica），数据此时以它为准。

第三步，拿到数据应该写到哪些 chunkserver 里之后，客户端会把要写的数据发给所有的 replica。不过此时，chunkserver 拿到发过来的数据后还不会真的写下来，只会把数据**放在一个 LRU 的缓冲区里**。

第四步，等到所有次副本都接收完数据后，客户端就会发送一个写请求给到主副本。我在上节课一开始就说过，GFS 面对的是几百个并发的客户端，所以主副本可能会收到很多个客户端的写入请求。主副本自己会给这些请求排一个顺序，确保所有的数据写入是有一个固定顺序的。接下来，主副本就开始按照这个顺序，把刚才 LRU 的缓冲区里的数据写到实际的 chunk 里去。

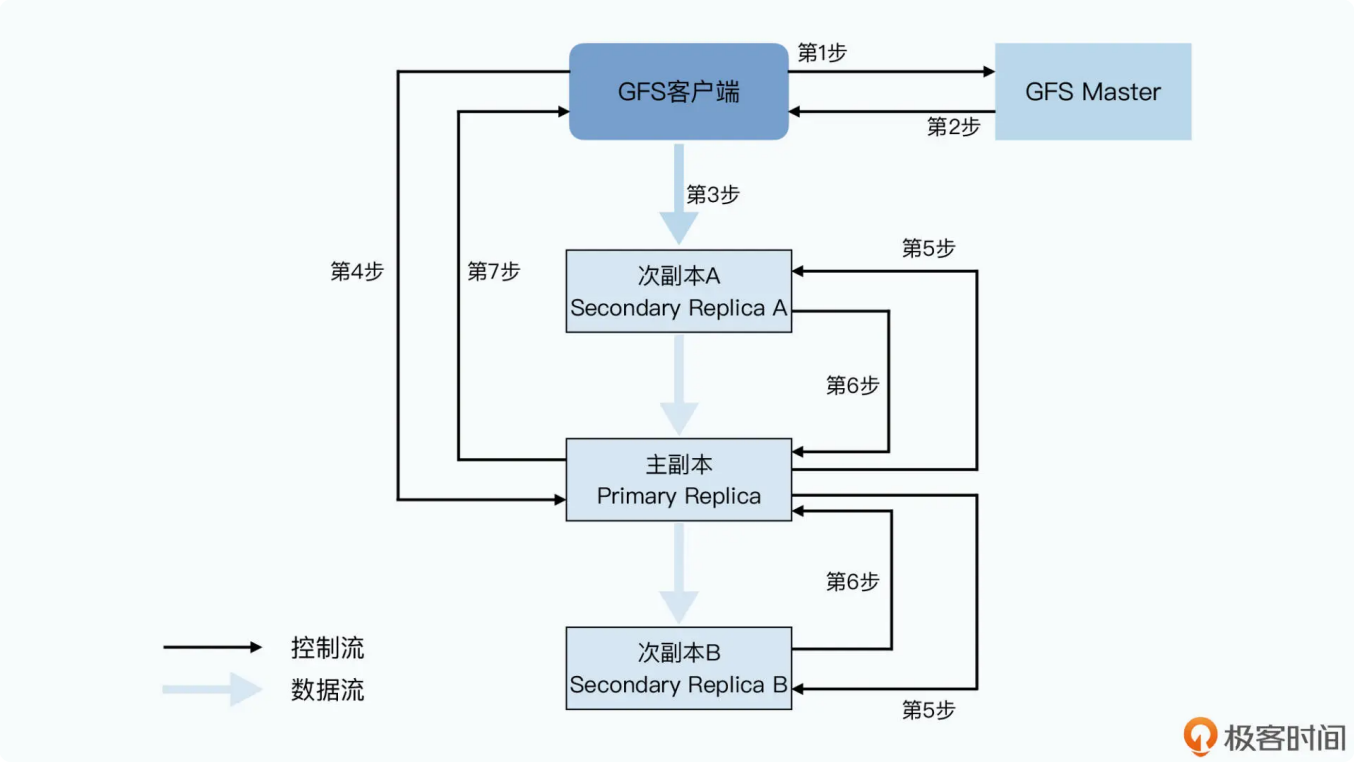
第五步，主副本会把对应的写请求转发给所有的次副本，所有次副本会和主副本以同样的数据写入顺序，把数据写入到硬盘上。

第六步，次副本的数据写入完成之后，会回复主副本，我也把数据和你一样写完了。

第七步，主副本再去告诉客户端，这个数据写入成功了。而如果在任何一个副本写入数据的过程中出错了，这个出错都会告诉客户端，也就意味着这次写入其实失败了。

所以在 GFS 的数据写入过程中，可能会出现主副本写入成功，但是次副本写入出错的情况。在这种情况下，客户端会认为写入失败了。但是这个时候，同一个 chunk 在不同

chunkserver 上的数据可能会出现不一致的情况，这个问题我们会放到下一讲来深入讨论。



GFS的数据写入采用了控制流和数据流分离的方式

我在这里也放了一张图，来帮助你理解这个数据写入的流程。从这张图上你会发现，GFS 的数据写入使用了两个很有意思的模式，来解决这节课一开始我提到的网络带宽的瓶颈问题。

分离控制流和数据流

第一个模式是控制流和数据流的分离。

和之前从 GFS 上读数据一样，GFS 客户端只从 master 拿到了 chunk data 在哪个 chunkserver 的元数据，实际的数据读写都不再需要通过 master。另外，不仅具体的数据传输不经过 master，后续的数据在多个 chunkserver 上同时写入的协调工作，也不需要经过 master。

这也就是说，控制流和数据流的分离，不仅仅是数据写入不需要通过 master，更重要的是实际的数据传输过程，和提供写入指令的动作是完全分离的。

其次，是采用了**流水线（pipeline）**式的网络传输。数据不一定是先给到主副本，而是看网络上离哪个 chunkserver 近，就给哪个 chunkserver，数据会先在 chunkserver 的缓冲区里存起来，就是前面提到的第 3 步。但是写入操作的指令，也就是上面的第 4~7 步，则都是由客户端发送给主副本，再由主副本统一协调写入顺序、拿到操作结果，再给到客户端的。

流水线式的网络数据传输

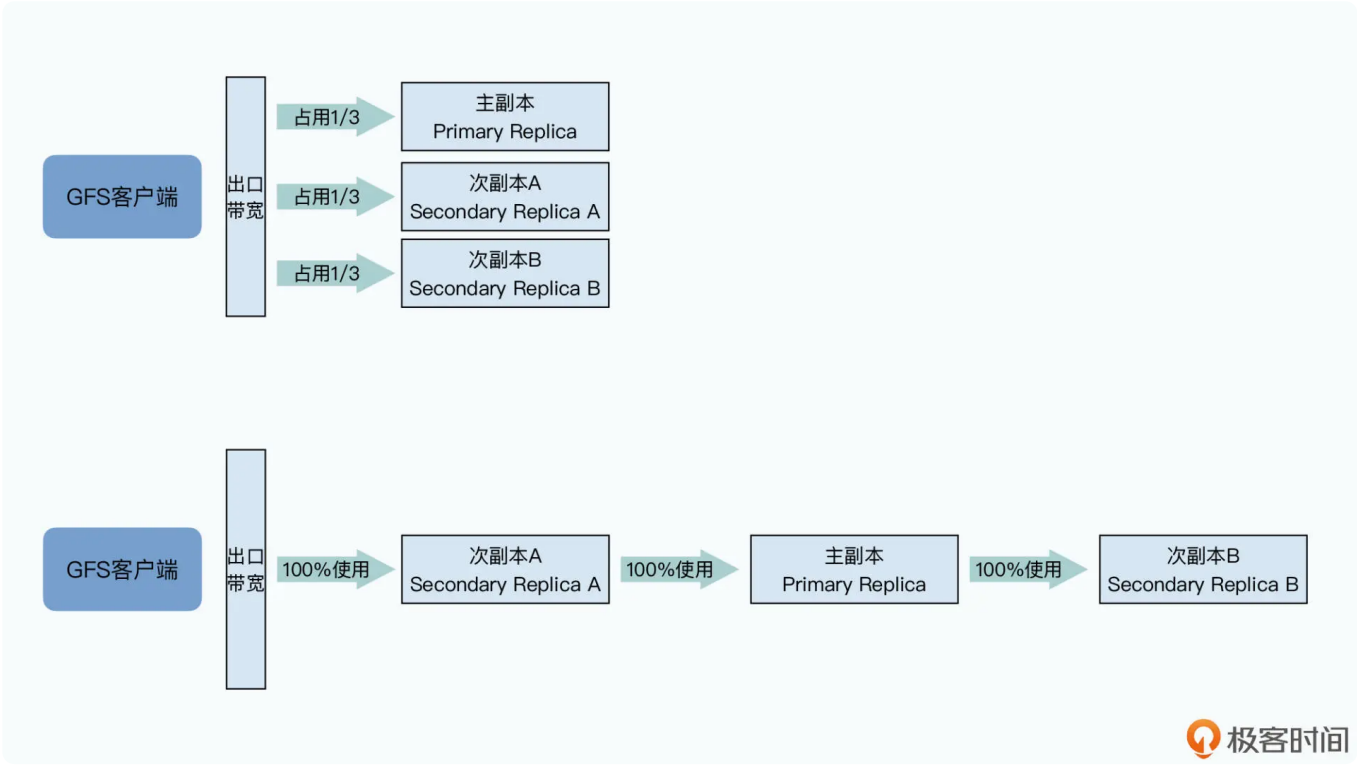
之所以要这么做，还是因为 GFS 最大的瓶颈就在网络。如果用一个最直观的想法来进行数据传输，我们可以把所有数据直接都从客户端发给三个 chunkserver。

但是这种方法的问题在于，**客户端的出口网络会立刻成为瓶颈**。

比如，我们要发送 1GB 的数据给 GFS，客户端的出口网络带宽有 100MB/ 秒，那么我们只需要 10 秒就能把数据发送完。但是因为三个 chunkserver 的数据都要从客户端发出，所以要 30s 才能把所有的数据都发送完，而且这个时候，三个 chunkserver 的网络带宽都没有用满，各自只用了 1/3，网络并没有被有效地利用起来。

而在流水线式的传输方式下，客户端可以先把所有数据，传输给到网络里离自己最近的次副本 A，然后次副本 A 一边接收数据，一边把对应的数据传输给到离自己最近的另一个副本，也就是主副本。

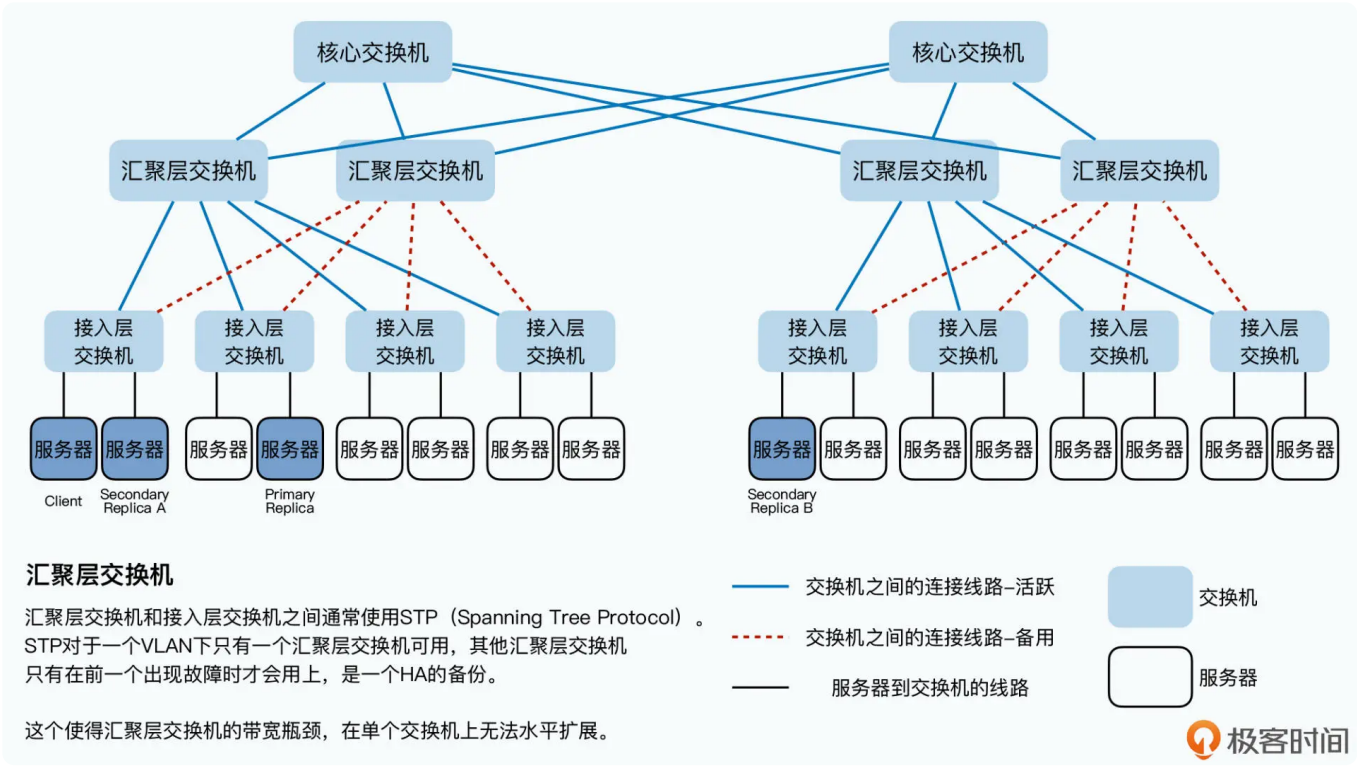
同样的，主副本可以如法炮制，把数据也同时传输给次副本 B。在这样的流水线式的数据传输方式下，只要网络上没有拥堵的情况，只需要 10 秒多一点点，就可以把所有的数据从客户端，传输到三个副本所在的 chunkserver 上。



流式传输的方式可以有效利用满客户端和chunkserver的网络带宽

不过到这里你可能要问了：**为什么客户端传输数据，是先给离自己最近的次副本 A，而不是先给主副本呢？**

这个问题，也和数据中心的实际网络结构有关，你可以先看看下面这张数据中心的网络拓扑图。



传统的三层交换的数据中心拓扑图

要知道，我们几百台服务器所在的数据中心，一般都是通过三层交换机连通起来的：

同一个机架（Rack）上的服务器，都会接入到一台**接入层交换机**（Access Switch）上；

各个机架上的接入层交换机，都会连接到某一台**汇聚层交换机**（Aggregation Switch）上；

而汇聚层交换机，再会连接到多台**核心交换机**（Core Switch）上。

那么根据这个网络拓扑图，你会发现，两台服务器如果在同一个机架上，它们之间的网络传输只需要通过接入层的交换机即可。在这种情况下，除了两台服务器本身的网络带宽之外，它们只会占用所在的接入层交换机的带宽。

但是，如果两台服务器不在一个机架，乃至不在一个 VLAN 的情况下，数据传输就要通过汇聚层交换机，甚至是核心交换机了。而如果大量的数据传输，都是在多个不同的 VLAN 之间进行的，那么汇聚层交换机乃至核心交换机的带宽，就会成为瓶颈。

所以我们再回到之前的链式传输的场景，GFS 最大利用网络带宽，同时又减少网络瓶颈的选择就是这样的：

首先，客户端把数据传输给离自己“最近”的，也就是在同一个机架上的次副本 A 服务器；

然后，次副本 A 服务器再把数据传输给离自己“最近”的，在不同机架，但是处于同一个汇聚层交换机下的主副本服务器上；

最后，主副本服务器，再把数据传输给在另一个汇聚层交换机下的次副本 B 服务器。

这样的传输顺序，就最大化地利用了每台服务器的带宽，并且减少了交换机的带宽瓶颈。而如果我们非要先把数据从客户端传输给主副本，再从主副本传输到次副本 A，那么同样的数据就需要多通过汇聚层交换机一次，从而就占用了更多的汇聚层交换机的资源。

独特的 Snapshot 操作

那么，在做了分离控制流和数据流，以及使用流水线式的数据传输方式之后，对于 GFS 的网络传输上，还有什么其他的优化空间吗？

你别说还真的有，那就是**为常见的文件复制操作单独设计一个指令**。

复制文件，相信这个是你用自己的电脑的时候，会常常做的事儿。在 GFS 上，如果我们用笨一点的办法，自然是通过客户端把文件从 chunkserver 读回来，再通过客户端把数据写回去。这样的话，读数据也经过一次网络传输，写回三个副本服务器，即使是流水线式的传输，也要三次传输，一共需要把数据在网络上搬运四次。

所以，GFS 就专门为文件复制设计了一个 Snapshot 指令，当客户端通过这个指令进行文件复制的时候，这个指令会通过控制流，下达到主副本服务器，主副本服务器再把这个指令下达到次副本服务器。不过接下来，客户端并不需要去读取或者写入数据，而是各个 chunkserver 会直接在本地把对应的 chunk 复制一份。

这样，数据流就完全不需要通过网络传输了。相信这个聪明的方法你也一定想到了。

小结

好了，通过这节课的学习，现在你对 GFS 是如何写入数据和复制文件应该就已经非常清楚了。那么，这里我们一起来回顾一下吧。

这节课，我先带你看了一下 GFS 的分布式集群的硬件配置，你会发现 2003 年 GFS 的最大的硬件瓶颈就是在网络。于是，在 GFS 设计数据写入机制的时候，就是经过仔细分析后针对这个问题来设计的。你可以重点关注以下这三个要点。

第一，和读数据一样，GFS 采用了控制流和数据流分离的方式。在写入数据的时候，客户端只是从 master 拿到 chunk 所在位置的元数据，而在实际的数据传输过程中，并不需要 master 参与，从而就避免了 master 成为瓶颈。

第二，在客户端向多个 chunkserver 写入数据的时候，采用了“就近”的流水线式传输的方案。这种方式，就尽可能有效地利用了客户端、chunkserver 乃至于交换机的带宽。

第三，对于常见的文件复制这个操作，GFS 专门设计了一个 Snapshot 的指令，针对文件复制，会直接在 chunkserver 本地进行，完全避免了网络传输。

其实，这三个动作，都不是什么理论上的创新，而是完全针对当时数据中心的网络架构、服务器硬件规格所进行的设计。也就是说，基于硬件设计，实际上不只是 GFS 的一个非常

重要的核心设计思想，它更是贯穿整个大数据系统领域的一个重要的主题。

推荐阅读

学完了这一讲之后，如果你对数据中心的网络架构有了一些设计思路，可以再去读一读 Facebook 工程团队在 2014 年写的这篇文章：Introducing data center fabric, the next-generation Facebook data center network。

而如果你觉得自己对于计算机的各类硬件性能不熟悉，我推荐你回头看一看我之前的《深入浅出计算机组成原理》的专栏，特别是其中关于存储器的部分。我们在后面的论文解读中，还会不断根据各种硬件特性和性能来思考我们的设计。

回头看起来，在“大数据”爆发之前，数据中心的数据流量通常是“南北大，东西小”，也就是大部分数据都是从某一台服务器，经过几层交换机，进入互联网，返回给终端用户。而在“大数据”爆发之后，数据中心的大量数据传输变成了数据中心的服务器横向之间的传输，而这个也让工程师们开始重新基于需求，重新设计数据中心需要的硬件和网络拓扑。那么，在学完今天这节课之后，我想你一定会对基于硬件性能设计系统，有更多的思考。

思考题

最后我想说，学习论文并不是背诵，重要是得总结和思考。所以我给你留了一道思考题，欢迎你和其他同学一起讨论一下。

在你接触过的系统和代码中，有没有什么设计，也是深度考虑了实际的硬件性能和瓶颈的呢？你可以留言说说，我们共同交流、互相进步。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 8 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 03 | The Google File System （一）： Master的三个身份

下一篇 05 | The Google File System （三）： 多写几次也没关系

精选留言 (12)

写留言

峰

2021-09-27

mysql利用b+出度打，层级底的特性，尽可能减少一次查询中随机io开销。
kafka利用磁盘顺序写入较随机写入快的特性，批量顺序写文件。
redis ignite 等内存数据库都基于内存性能远胜于磁盘等持久化外部存储，从而基于内存做存储系统。

展开

作者回复: 👍

8

webmin

2021-09-27

今天课程中关于网络优化的内容，基本是出自GFS论文中的3.2节Data Flow，我很好奇是因为老师有关于广告系统的开发经验，所以能从一个300个单词左右的小节中看出这么丰富的信息，还是老师有其它的分析框架或辨识方法？还望老师抽时间传授。

另加一个注解流水线（pipeline）式的网络传输是有效利用了网络是全双工的原理，即左...

展开

作者回复:
论文大部分都是很精简的，所以如果每段内容彻底弄清楚其实展开都有很多问题可以讲，而且GFS的核心瓶颈往往就在网络，所以我特地深入展开一下。

关于网络优化，更多是因为过去的经历和经验中遇到很多这方面的困扰吧。因为十年前是没有云计算平台可以用的，需要自己来建数据中心。跨地域的数据中心的网络传输也要想办法自己解决，所以的确会遇到各种应用开发遇不到，但是教科书和论文里遇到的问题。

5

Ping

2021-09-27

能再解释下"南北大，东西小"是什么意思吗？

展开

作者回复: Ping同学，

你好，你看我在里面放的数据中心的网络架构图。按照上北下南，左西右东来理解方位。

在有大数据需求之前，一般我们的网络流量，都是互联网请求访问服务器，然后通过三层交换机到达某一台具体服务器，服务器给出结果，再通过三层交换机返回给外部。这里面的网络流量都是南北向的。

但是在大数据处理过程中，大部分数据传输都是最下面一层服务器之间互相传输，也就是东西向的。

2

音速起子代购

2021-10-06

请问一个问题，，chunk的副本位置关系不是由master掌握吗？那在复制过程中，由主副本如何转发通知次副本消息呢？主副本如何知道次副本的位置？

展开

火娃儿

2021-10-05

HBASE 就是 利用LSM ,用一定的 归并 门限来 逐步 合并 每次 写操作，达到门限再进行 写磁盘操作。

Lebron

2021-10-02

老师，你贴的那篇文章我无法访问呢，有没有国内的链接，像你之前课程的那种PDF的链接？

leslie

2021-09-30

UDP替代TCP去实现网络的传输，传完了就好了；监控而已-丢失了再传即可；不过当时忽略了一个关键问题-windows并不适合去用很多适合在linux下的软件，导致了备份和恢复的

灾难性隐患。

作者回复: Windows现在有了WSL2了，体验还是不错的。

Geek_2e6a7e
2021-09-28

阿里自研MaxCompute大规模集群计算有什么特别创新的地方么，这块有相关资料或者论文参考下么？

作者回复: 抱歉，我没有使用或者研究过MaxCompute。不过我相信市场上主流的大数据方案其实底层逻辑都是相同的。

我简单看了一下MaxCompute的介绍，我觉得MaxCompute更像一个“产品”而不是一种专门的“技术”，是对多种数据需求，包装成了一个整体的产品供云计算的用户使用。

zhanyd
2021-09-28

适合自己的就是最好的，不一定要去追求什么高大上的技术，能够低成本满足需求的就是好方案。创新不一定是要创造出什么新东西， 把一些东西按适当的方式组合在一起也是创新。

作者回复:

陈迪
2021-09-28

思考一个问题，20年过去了，硬件环境已经哪些发生了根本性的变化？现代的分布式文件系统应该什么样的？

展开

作者回复: 好问题，值得探讨。

从大数据系统来看，随着SSD的廉价，原先觉得SSD不适合作为Hadoop的存储层这一点在逐渐失效。SSD，机械硬盘，乃至磁带冷备随着数据越来越多，价格越来越便宜，成为各有应用场景的

硬件了。

SSD的出现使得随机读的性能上了几个数量级，不过这个针对的更多是数据系统的Serving层。

不过网络瓶颈似乎变化不大。

在路上
2021-09-27

徐老师好，GFS论文3.2 Data Flow中提到“Without networkcongestion, the ideal elapsed time for transferring B bytes to R replicas is B/T + RL where T is the network throughput and L is latency to transfer bytes between two machines. Our network links are typically 100 Mbps (T), and L is far below 1 ms. Therefore, 1 MB can ideally be distributed in about 80 ms.”...

展开

作者回复: 我们先要弄清楚什么是网速？我们关注的是延时还是吞吐量？

在pipeline传输里，A节点传输数据到B，B转手给到C，C再给D，B和C都不需要等数据传输完再往下发。而是收多少立刻再发多少。
所以时间就是 B/T + R个节点之间的延时（A->B，B->C，C->D 一共三个延时）T。

在路上
2021-09-27

徐老师好，分布式系统的提出就是因为单机硬件资源受限，需要将计算和存储任务交给多台计算机协同完成，传统的网络拓扑结构让网络资源不能像cpu、memory和disk一样分片，所以网络往往成为分布式系统的瓶颈。在多级分流系统中，常常分层处理网络流量，减少流量向深层节点传播。比如设计秒杀系统时，可以在第一层随机让一部分请求失败，防止流量把下游的系统打垮。在人脸识别系统中，可以利用边缘节点检查基本特征，做...

展开